#include<bits/stdc++.h>

using namespace std;

typedef long long LL;

const int maxn = 1e5 + 100;

int q = 0;

LL x = 0,y = 0,k = 0;

int main()

{

scanf("%d",&q);

while(q--)

{

scanf("%I64d %I64d %I64d",&x,&y,&k);

//x = max(x,y);y = min(x,y);//这犯了一个傻逼错误，x=max(x,y)后，x的值可能就就改变了

//比如y > x,那么x=max(x,y)后，x等于y，这样y=min(x,y)后，y还是y，比没有变成最小值

if(y > x)

{//还是这种方法靠谱

swap(x,y);

}

if(x > k){//最小步数就是min(x,y)+abs(x-y),也就是max(x,y),注意这里x,y>0，所以不用考虑负数

//比赛的时候，我还傻傻地考虑了负数

printf("-1\n");

continue;

}

if((x - y) & 1)//是奇数

{

/\*

有一种情况当你在x-y这一段的时候，如果(x-y)是偶数，那么可以全部都走对角线

比如(0,0)->(0,2)可以(0,0)->(-1,1)/(1,1)->(0,2)，采取这种走位就最好。如果(x-y)是奇数，那么

剩余的那一步可以有大用处，如果先走(x-y)步，然后走正方形，如果剩余的步数k-(x-y)走完对角线

剩余的k-(x-y)-y是偶数，那么反复走对角线即可，但是如果是奇数，那么多余的那一步，可以在走(x-y)

那里剩余的一步，比如(0,1)->(0,2)的一步可以变成(0,1)->(-1,1)->(0,2),这样的话就只有(0,1)->(-1,1)

那一步不是对角线，所以k-=1;

所以，x-y剩余的那一步可以起到调节正方形剩余步数的功能

\*/

k -= 1;

}

else if((k - x) & 1)//k-x代表走完正方形后到走到(n,m)还能用的步数

{

/\*

如果没有那个关键的一步，那么(x-y)全部都是对角线，走完正方形剩余的步数如果是偶数，则全是对角线,

如果是奇数，那么要走一个一横一竖，代替一个对角线，刚好把那个多余的一步抵消掉，这个时候要减去2

\*/

k -= 2;

}

printf("%I64d\n",k);

}

return 0;

}